

## Tutorial 10 - Creació d'un joc (II)

### Pas 1 de 22

Continuem amb el joc que hem començat en el tutorial anterior, anem a completar el desenvolupament afegint-hi obstacles que la nau ha d'evitar.

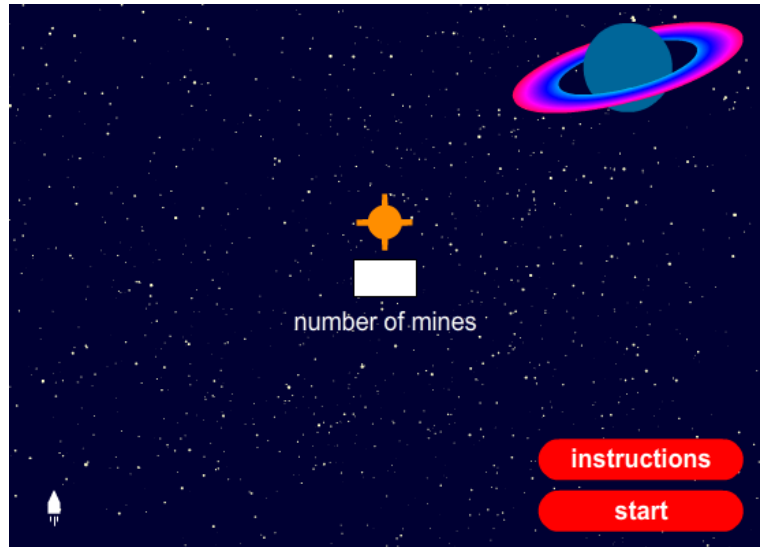
En primer lloc, per als obstacles crearem una animació utilitzant una interpolació de forma.

Aprendrem a afegir objectes dinàmicament des de la biblioteca, a més de posicionar-los a l'escenari aleatòriament.

També aprendrem a generar moviment aparentment erràtic.

Farem servir sons de la biblioteca sense necessitat d'afegir-los a l'escenari.

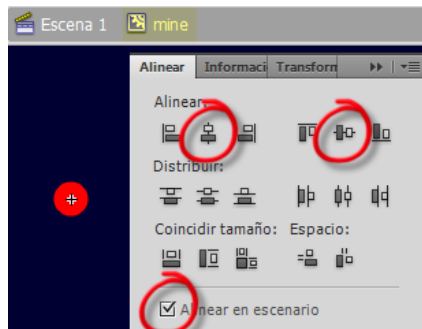
Finalment, permetrem al jugador triar el nombre d'obstacles que apareixeran en el joc utilitzant un camp d'introducció de text.



### Pas 2 de 22

En primer lloc, crearem els obstacles que haurà d'esquivar la nau. Seleccionem **Insertar > Nuevo símbolo**. Donem al símbol el nom *mine* i com a tipus seleccionem **Clip de película**.

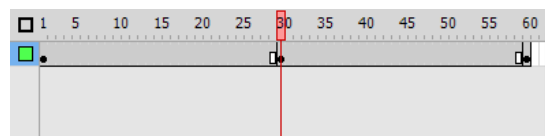
En la línia de temps d'aquest nou clip, dibuixarem un cercle vermell petit de 25 x 25 px, sense traç i amb farciment vermell. Amb el panell **Alinear**, centrem aquest cercle al seu escenari.



Aquest clip tindrà una animació senzilla de la qual canviarem la forma i el color, per després tornar-la a la forma original.

L'animació durarà 60 fotogrames. Inserim un **fotograma clau** premen **F6** en el *fotograma 60*. D'aquesta manera es copiarà el contingut del fotograma 1 en el fotograma 60.

Inserim un altre **fotograma clau** en el *fotograma 30*. També es copiarà el contingut del fotograma 1, però en aquest cas la intenció és modificar en aquest fotograma la forma del cercle original.

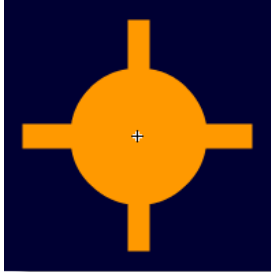


### Paso 3 de 22

Amb el fotograma clau 30 seleccionat, dibuixem dos rectangles, un de vertical i un altre d'horitzontal, que centrarem a mesura que els dibuixem. Podem treballar amb zoom per a més comoditat.

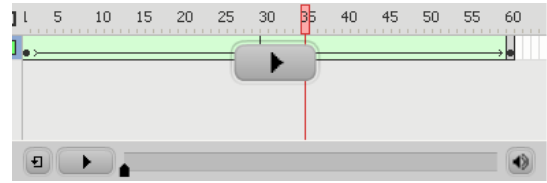
Aquests rectangles tampoc tindran traç i tindran com a farciment el mateix color vermell que el cercle. D'aquesta manera, els rectangles quedaran units al cercle en una sola forma.

Per acabar, canviem el color de la forma composta a un to ataronjat.



Per a crear una transició des de la forma del cercle original al cercle amb els rectangles, i una altra transició fins a tornar de nou a la forma i color originals, hem de crear interpolacions de forma.

Per a això fem clic amb el **botó dret** del ratolí sobre l'àrea grisa que hi ha entre dos fotogrames clau, i seleccionem el menú contextual **Crear interpolación de forma**. Repetim el procés per a la transició compresa entre els altres dos fotogrames clau.



Podem crear interpolacions de forma entre dues formes qualssevol. Per a controlar canvis de forma complexos, podem utilitzar els consells de forma, encara que en el nostre cas no caldrà. Consulta l'ajuda del Flash referent a això si has de crear interpolacions de forma complexes.

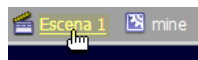
### Pas 4 de 22

Si premem Intro o movem el cap lector, veurem que la forma canvia del cercle vermell inicial, a la forma composta ataronjada, per a tornar posteriorment al cercle vermell.



Podem experimentar amb diferents formes o colors fins a crear una interpolació de la manera que ens agradi més.

Amb això donem per finalitzada la creació de la mina, per la qual cosa tornem a l'escena principal.

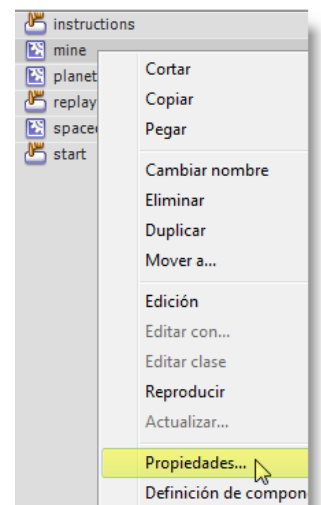


A continuació afegirem còpies a l'escenari del clip *mine* utilitzant ActionScript.

Fins ara, per a poder fer referència a un objecte utilitzàvem en la programació el seu nom d'instància. Per a assignar un nom d'instància a un clip, seleccionàvem el clip en l'escenari i escrivim el seu nom d'instància en l'inspector de *Propiedades*.

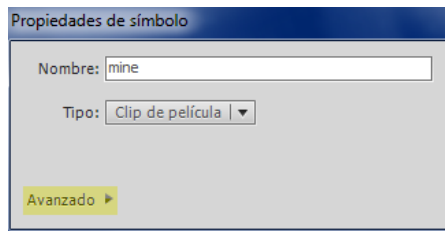
En cas que l'objecte no sigui en l'escenari, l'hem d'exportar per a ActionScript des de la biblioteca.

Fem clic amb el **botó dret** del ratolí sobre el nom del clip (*mine* en aquest cas), i seleccionem **Propiedades** en el menú contextual.

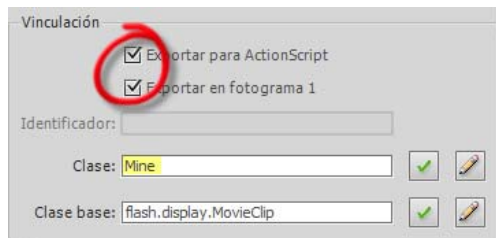


## Paso 5 de 22

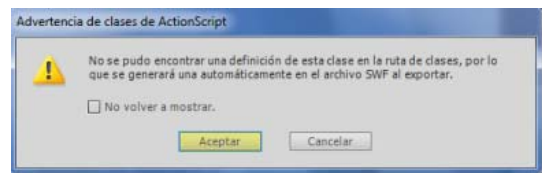
En la finestra **Propiedades de símbolo**, premem el botó **Avanzado**.



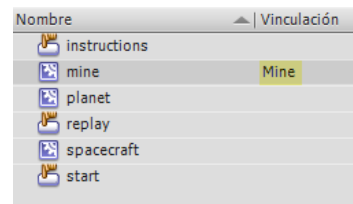
En l'àrea **Vinculació** marquem les caselles **Exportar para ActionScript** i **Exportar en fotograma 1**. Escrivim el nom de classe *Mine* (per convenció els noms de les classes comencen amb majúscula).



Quan premem **Aceptar** ens apareixerà una finestra d'advertiment. Tornem a prémer **Aceptar** perquè es generi de manera automàtica la definició de la classe *Mine*.



Podem comprovar que en el panell **Biblioteca**, a l'àrea de **Vinculació**, apareix el nom de classe que hem assignat al clip. Podríem haver posat també el nom de vinculació directament des d'aquest panell, i el clip s'exportaria igualment.



Després d'haver exportat el clip en una classe anomenada *Mine*, ja podem fer referència a aquesta classe en la nostra programació.

## Pas 6 de 22

Definirem una funció nova que anomenarem `putMines`.

```
function putMines():void
{
    var mine:Mine;
    mine = new Mine;
    mine.x = 275;
    mine.y = 200;
    stage.addChild(mine);
}
```

En la primera línia d'aquesta funció, definim que `mine` és un objecte de la classe *Mine*. Després creem una instància d'aquesta classe, que serà una còpia nova del clip que tenim en la biblioteca. Després d'aquesta definició, ja podem fer referència a la nova instància que hem creat.

En les línies següents, definim que la posició de l'objecte sigui el centre de l'escenari (`x:275` e `y:200`). Finalment, afegim aquest objecte a l'escenari (`stage`) amb el mètode `addChild`.

Perquè aquesta funció s'executi, en algun moment l'hem de cridar amb la instrucció `putMines()`. Escriurem aquesta instrucció dins de la funció `playGame`, ja que serà el moment en què volem que s'afegeixin les mines a l'escenari.

Si provem la pel·lícula, comprovarem que en prémer el botó `start` (que crida la funció `playGame`, que al seu torn crida la funció `putMines`), apareix una mina en el centre de l'escenari.

Perquè la posició en la qual apareix la mina sigui aleatòria, utilitzarem el mètode `random` de la classe `Math`.

`Math.random()` torna un nombre aleatori comprès entre 0 i 1. Si multipliquem aquest valor aleatori per 550 (l'amplada de l'escenari), el nombre que tornarà estarà comprès entre 0 i 550.

Per tant, per a crear una posició aleatòria de la mina dins de l'escenari, que mesura 550 x 400, les línies que determinen la posició del clip dins de la funció `putMines` quedaran com s'indica a continuació:

```
mine.x = Math.random() * 550;
mine.y = Math.random() * 400;
```

La posició `x` del clip serà un nombre comprès entre 0 i 550, mentre que la posició `y` serà un nombre comprès entre 0 i 400. Cada vegada que s'executi la funció `Math.random()`, el nombre tornat serà diferent.

## Paso 7 de 22

Per ara només s'afegeix una còpia del clip a l'escenari. Per a afegir diverses còpies, hem de fer un bucle que creï diverses mines i les afegeixi a l'escenari.

Juntament amb les variables `course` i `speed`, definirem una nova variable anomenada `numMines`, el valor del qual serà el nombre de mines que volem crear (per exemple, 10).

```
var numMines:Number = 10;
```

Ara farem servir un bucle `for` per a crear aquestes 10 còpies de la mina.

Els bucles `for` tenen l'estructura següent:

```
for (initial value; conditional statement; expression that changes the value)
{
    //statements
}
```

Per exemple, en el nostre cas:

```
for (var i:Number = 0; i < numMines; i++)
{
    //statements
}
```

Aquest bucle funcionaria de la manera següent:

- Creem una variable anomenada `i` amb un valor inicial de 0.
- Comprovem si es compleix la condició, que en aquest cas és que el valor d'`i` sigui més petit que el valor de `numMines`.
- En complir-se la condició que `i < numMines`, executarem les sentències que hi hagi entre les claus del bucle `for`.
- Augmentem el valor d'`i` en una unitat (`i++` significa `i = i + 1`).
- Tornem a comprovar la condició. Ara `i` val 1, que continua essent més petit que 10 (valor de `numMines`).
- Atès que la condició es continua complint, tornem a executar les sentències, i sumem una altra unitat a `i`, que ara valdrà 2.
- Quan `i` valgui 10, moment en què no es complirà que `i` sigui més petit que `numMines`, ja no s'executarà el bucle. En començar amb un valor d'`i` = 0, el bucle s'haurà executat un total de 10 vegades.

## Pas 8 de 22

Per tant, per crear la quantitat de mines que hàgim indicat en la variable `numMines`, la funció `putMines` quedarà com s'indica a continuació:

```
function putMines():void
{
    var mine:Mine;
    for (var i:Number = 0; i < numMines; i++)
    {
        mine = new Mine;
        mine.x = Math.random() * 550;
        mine.y = Math.random() * 400;
        stage.addChild(mine);
    }
}
```

Si ara provem la pel·lícula, podem comprovar que apareixen 10 mines en l'escenari.

Totes les mines fan la seva animació alhora. Podem fer que cada mina comenci l'animació en un fotograma diferent, i que porti el seu cap lector a un fotograma aleatori entre l'1 i el 60, que és el nombre de fotogrames que té l'animació.

Per tant, aquesta vegada necessitem generar un nombre aleatori entre 1 i 60. En aquest cas el nombre tomat ha de ser enter.

Si utilitzem `Math.random() * 60` obtindrem nombres decimals entre 0 i 60. Per a assegurar-nos que el nombre resultant sigui un enter comprès entre 1 i 60 podem utilitzar el mètode `ceil`, que arrodoneix a l'alça un nombre decimal.

Per tant, dins del bucle `for`, després d'haver determinat una posició aleatòria per a cada mina i abans d'afegir la mina a l'escenari, escriurem la instrucció:

```
mine.gotoAndPlay(Math.ceil(Math.random() * 60));
```

Tornem a provar la pel·lícula. Ara, en començar el joc, es creen 10 còpies de la mina, i cada una comença l'animació en moments diferents.

## Paso 9 de 22

Per convertir les mines en obstacles contra els quals hem d'evitar xocar, afegirem un detector a cada mina, perquè avalui en cada moment si està xocant amb la nau.

Per a això afegim en primer lloc un listener per a cada mina en el mateix bucle que utilitzem per a crear-les, dins, per tant, de la funció `putMines` i dins del bucle `for`, i abans d'afegir les mines a l'escenari amb `addChild`.

```
mine.addEventListener(Event.ENTER_FRAME, enemy);
```

La nova funció, que hem anomenat `enemy`, comprovarà si cada mina a la qual hem afegit el listener xoca amb la nau en algun moment. Si hi xoca, llavors executarem la funció `gameOver`, amb el paràmetre `"lose"`.

```
function enemy(e:Event):void
{
    if (e.target.hitTestObject(spacecraft_mc))
    {
        gameOver("lose");
    }
}
```

Si ara provem el nostre joc, tant si guanyem com si perdem, ens apareixerà un error perquè, malgrat que no hi ha cap nau, la funció `enemy` continuarà comprovant si cada mina xoca amb la nau. En no trobar cap nau en l'escenari, es mostrarà l'error.

Per tant, el primer que hem de fer en la funció `gameOver`, abans de l'ordre d'anar a un altre fotograma en el qual no hi hagi la nau, és eliminar els listeners que hem afegit a les mines. Alhora que eliminem cada listener, aprofitarem per eliminar les mines de l'escenari. En el pas següent mostrarem com fer-ho.

L'escenari (`stage`) funciona com un contenidor que conté en primer lloc la línia de temps, i que també conté cada mina que li hem anat afegint amb `addChild`. Després de crear totes les mines, l'escenari tindrà 11 elements secundaris. El primer, en el nivell inferior de visualització (nivell 0), serà la línia de temps. Després hi haurà totes les mines, cada una en un nivell superior a l'anterior.

La quantitat de mines creades depèn del valor que hàgim donat a `numMines`. Per tant, l'última mina creada es mostrarà en un nivell de visualització que coincideix amb `numMines`.

## Pas 10 de 22

Al començament de la funció `gameOver`, abans de les sentències en les quals eliminàvem els listeners de la nau, podem eliminar el listener de cada mina, i després eliminar cada mina, amb el codi següent:

```
for (var i:Number = numMines; i > 0; i--)
{
    stage.getChildAt(i).removeEventListener(Event.ENTER_FRAME,
    enemy);
    stage.removeChild(i);
}
```

Analitzem el que fa aquest bucle. El bucle començarà amb un valor d'`i` igual a `numMines`, que en el nostre cas és 10. Com que la variable `i` és més gran que 0, s'executaran les sentències que es troben entre les claus.

Després reduïrem en una unitat el valor d'`i` (`i--` significa `i = i - 1`). Per tant, `i` valdrà 10 en la primera execució del bucle, 9 en la volta següent, etc., fins a l'últim valor que compleix la condició, és a dir, fins que `i` valgui 1 (compleix `i > 0`).

En les sentències que hi ha dins del bucle, en primer lloc eliminem el listener de l'element que hi ha dins de l'`stage`, en el nivell de visualització `i` (10, 9, 8, 7, ..., 1), que en el nostre és cada mina que hem creat. D'aquesta manera ja no tractaran de detectar possibles col·lisions amb la nau. Amb la línia següent eliminem la mina a la qual acabem d'eliminar el seu listener.

Ara el nostre joc serà funcional i no tindrà errors, però introduïrem millores en els passos següents:

- En primer lloc, ens assegurarem que la posició inicial de les mines no sigui gaire a prop de la nau, perquè ens doni temps a començar a jugar.
- Per fer el joc més complex, afegirem un moviment aleatori de les mines per l'escenari.
- Canviarem la imatge del planeta per la imatge d'una mina en el fotograma `lose`.
- Després afegirem un so per a quan guanyem i un altre per a quan perdem.
- Finalment, permetrem a l'usuari decidir el nombre de mines que vol que apareguin en el joc.

## Paso 11 de 22

Comencem per la primera millora.

A causa de l'aleatorietat de la posició de les mines, és possible que fins i tot abans de moure la nau ja hi hagi alguna mina amb la que hi xoqui.

Per solucionar aquest problema, substituïrem el codi en el qual creàvem la posició `x` i `y` de cada `mine`, dins de la funció `putMines`, per aquest codi:

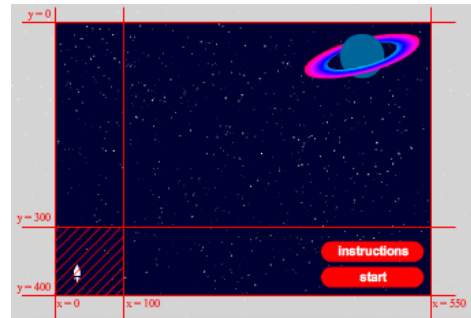
```
do
{
    mine.x = Math.random() * 550;
    mine.y = Math.random() * 400;
} while (mine.x < 100 && mine.y > 300);
```

La sentència `do...while` el que fa és executar en primer lloc el que hi hagi entre claus. Mentre es compleixi la condició que hi ha entre els parèntesis de `while`, es tornarà a executar el que hi hagi entre les claus del `do`.

En aquest cas, creem una posició a l'atzar per a la mina. Si la posició és a prop de la cantonada inferior esquerra (si la posició horitzontal de la mina és més petita que 100, i a més la seva posició vertical és més gran que 300), llavors tornarà a generar-se una posició aleatòria per a la mina.

Aquest procés es repetirà fins que la condició no es compleixi, cosa que significarà que la posició generada per a la mina ja no és a prop de la nau.

D'una manera més gràfica, si la posició aleatòria de la mina és en la zona del quadrat ratllat, llavors canviarem la posició de la mina de manera aleatòria, fins a trobar una posició vàlida.



## Pas 12 de 22

Afegirem el moviment de vibració de les mines en l'escenari. Creem una variable nova a l'inici de la programació, juntament amb les variables `course`, `speed` i `numMines`, que anomenarem `vibration`. Hi assignem un valor de 5.

```
var vibration:Number = 5;
```

Programarem que la posició de la mina varii, des de la posició en la qual es troba en cada moment, a una nova posició aleatòria que sigui a una distància màxima de 5 píxels de la posició actual, que és la quantitat que hem assignat a la variable `vibration`.

És a dir, si la posició actual d'una mina és `x=200` i `y=300`, a l'instant següent el valor d'`x` estaria entre 195 i 205, mentre que el valor d'`y` podria estar entre 295 i 305. D'aquesta manera, la mina vibrarà un màxim de 5 píxels en cada sentit (horitzontal i vertical).

Tal com hem après anteriorment, la sentència

```
Math.random() * vibration
```

tornarà un valor entre 0 i 5 (que és el valor que hem donat a la variable `vibration`).

Per tant, la sentència

```
Math.random() * vibration - Math.random() * vibration
```

tornarà un nombre entre -5 (0-5) i 5 (5-0).

Dins de la funció `enemy` (que s'executa cridada per un `ENTER_FRAME` de cada `mine`), però fora del condicional que avalua si hi ha xoc entre la mina i la nau, escriurem el codi següent:

```
e.target.x += Math.random() * vibration - Math.random() * vibration;
e.target.y += Math.random() * vibration - Math.random() * vibration;
```

Per evitar que amb aquesta vibració aleatòria la mina pugui acabar fora de l'escenari, dins de la mateixa funció afegirem unes sentències condicionals que avaluin si la mina és fora dels límits, i si hi és la tornarem a col·locar en el límit de l'escenari.

```
if (e.target.x < 0)
{
    e.target.x = 0;
}
```

Aquest condicional evitarà que la nau surti pel costat esquerre de l'escenari. Hem d'afegir-hi altres condicionals similars per a la resta dels casos (si `x > 550` llavors que `x = 550`, si `y < 0` llavors `y = 0`, i si `y > 400` llavors `y = 400`).

## Paso 13 de 22

Provem la pel·lícula per comprovar si s'efectua la vibració de les mines. Veurem que encara que les mines mai no desapareixen completament de l'escenari, pot ser que en alguns moments vegem mitja forma de la mina fora de l'escenari, perquè el punt de registre de la mina és en el seu centre.

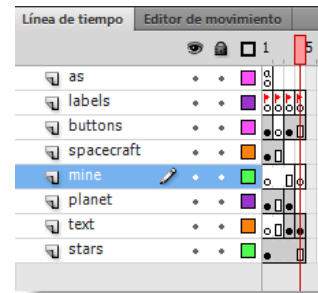
Si volem, podem modificar els límits de moviment de la mina canviant els valors dels condicionals del pas anterior. Per exemple, podem reduir l'espai en el qual poden moure's les mines en 25 píxels en cada banda.

La millora següent va a ser eliminar el planeta en el quart fotograma (fotograma *lose*), i mostrar com a substitució una mina en el centre de l'escenari.

En primer lloc, eliminem el quart fotograma de la capa *planet* prement-lo amb el **botó dret** del ratolí, i seleccionant **Quitar fotogramas**.

Afegim una nova capa que anomenarem *mine*. Situem aquesta capa per sobre de la capa *planet*.

Després premem amb el **botó dret** del ratolí el quart fotograma de la capa *mine*, i seleccionem **Insertar fotograma clave**.



En aquest nou fotograma clau, arrosseguem una instància del clip *mine* des de la biblioteca fins al centre de l'escenari, i n'augmentem la mida fins, per exemple *AN:100* i *AL:100*.

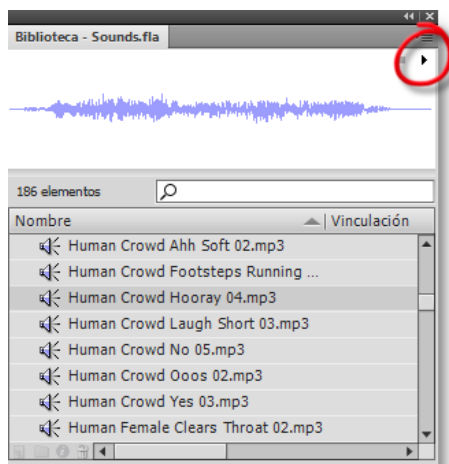


## Pas 14 de 22

Buscarem sons que associarem al moment de guanyar o de perdre la partida.

A **Ventana > Bibliotecas comunes > Sonidos** podem trobar alguns exemples de sons que podem utilitzar en els nostres projectes.

Per escoltar els diferents sons, seleccionem el so en la biblioteca i premem el botó petit per reproduir el que hi ha en la finestra on es mostra l'ona del so.



Utilitzarem per exemple els sons *HumanCrowdHooray04.mp3* per al moment de guanyar, i *Multimedia Internet CD-RomFlash Blast06.mp3* per al de perdre.

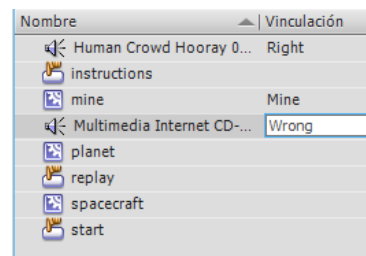
Podem ampliar la mida de la biblioteca de sons o l'àrea dels noms per a llegir els noms dels sons amb més comoditat.

Per utilitzar aquests sons, els podem arrossegar directament des de la biblioteca de sons fins a la biblioteca del nostre joc.

Una vegada incorporats en la nostra biblioteca, podem tancar la biblioteca de sons.

Igual com vam fer amb el clip de la mina, exportarem tots dos sons per a l'ActionScript.

Aquesta vegada premerem directament sobre l'àrea **Vinculació** del panell **Biblioteca**, assignant a un so el nom de classe *Right* i a l'altre so el nom de classe *Wrong*.



## Paso 15 de 22

Per utilitzar aquests sons afegirem, juntament amb la definició d'altres variables a l'inici de la programació, les línies següents:

```
var soundWin:Right = new Right();
var soundLose:Wrong = new Wrong();
```

La variable `soundWin` pertanyerà a la classe `Right` i serà una instància nova de aquest so. De la mateixa manera, `soundLose` serà un objecte de la classe `Wrong`.

Dins de la funció `gameOver` reproduïrem un so o un altre depenent de si hem guanyat o hem perdut.

En aquesta funció, la variable que ens indica si hem guanyat o perdut és `frameLabel`. Per tant, el que farem és comprovar si `frameLabel` té el valor "win". Si el té, reproduïrem `soundWin`. Si no el té, reproduïrem `soundLose`.

Per tant, dins de la funció `gameOver` inclourem aquest codi:

```
if (frameLabel == "win")
{
    soundWin.play();
}
else
{
    soundLose.play();
}
```

El signe d'igual doble (==) compara si `frameLabel` és "win". Aquest signe és per a comparar, mentre que un sol signe d'igual (=) es para assignar.

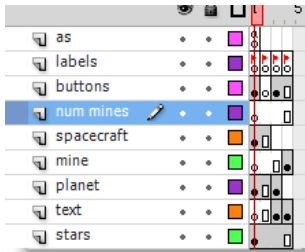
Si `frameLabel` és "win", es reproduirà `soundWin`. Si no ho és (else) es reproduirà `soundLose`.

## Pas 16 de 22

Ja tenim el joc pràcticament acabat. Podem augmentar els valors a les variables `speed`, `numMines` i/o `vibration` per a augmentar la dificultat del joc. Per exemple, una vibració de 15 complicaria considerablement el joc, ja que el moviment de les mines seria menys previsible.

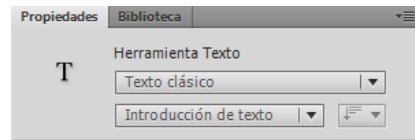
Podem fer que el mateix jugador modifiqui aquests valors amb un camp d'introducció de text. Com a exemple, permetrem que el jugador decideixi el nombre de mines amb què vol jugar.

Per fer-ho, en primer lloc creem una nova capa anomenada *num mines* i la situem sota la capa *buttons*.



Seleccionem l'eina **Texto**.

En l'inspector de *Propiedades* seleccionem **Texto clásico** i **Introducción de texto**.



Fem clic sobre qualsevol punt en l'escenari per crear un camp d'introducció de text en la capa *num mines*.

Seleccionem aquest camp de text acabat de crear. En l'inspector de *Propiedades*, li assignem el nom *mines\_txt*.

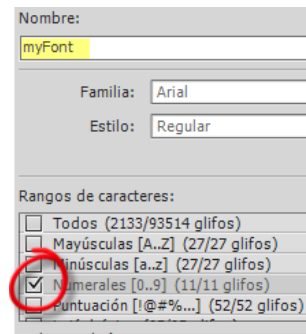


## Paso 17 de 22

Excepte en el cas en el qual utilitzem fonts del dispositiu, hem d'incorporar les fonts que utilitzarem en els camps de text dinàmic o d'introducció de text.

En l'àrea **Caràcter** seleccionem el tipus de lletra i la mida que vulguem, i fem clic sobre el botó **Incorporar**.

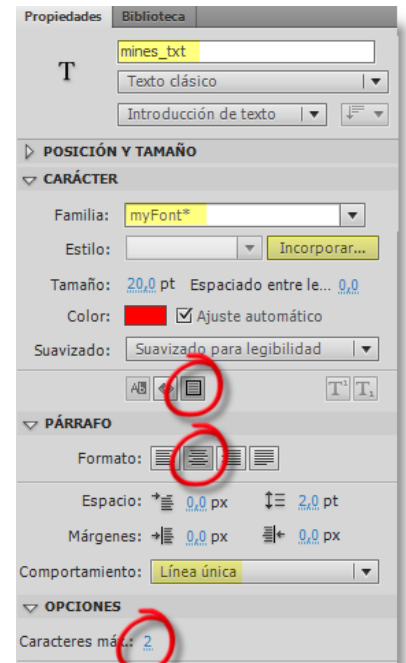
En la pantalla d'incorporació de fonts, escrivim el nom *myFont* per a aquesta font incorporada.



En Familia i Estilo apareixerà preseleccionat el que teníem en l'inspector de *Propiedades*.

En **Rangos de caracteres** seleccionem **Numerales**, ja que són els únics caràcters que necessitem.

Una vegada acceptada la pantalla d'incorporació de fonts, tornem a l'inspector de *Propiedades* i en Familia seleccionem en el desplegable la font que hem incorporat. El seu nom apareixerà per sobre de la llista de la resta de les fonts, i anirà seguit d'un asterisc per indicar que es tracta d'una font incorporada.



Seleccionem un **color** vermell a fi que ressalti sobre el fons, i marquem la casella **Mostrar borde alrededor del texto**.

Això farà que aparegui un requadre blanc amb un marc negre. El marc negre amb prou feines el veurem, ja que el nostre joc té un fons molt fosc, però el color blanc del requadre ens mostrarà clarament on hi ha el nostre camp de text.

En l'àrea **Pàrrafo**, alineem el text en el **centre** i triem un comportament de **Línea única**. Finalment, en l'àrea **Opciones** seleccionem que es puguin escriure un màxim de 2 caràcters.

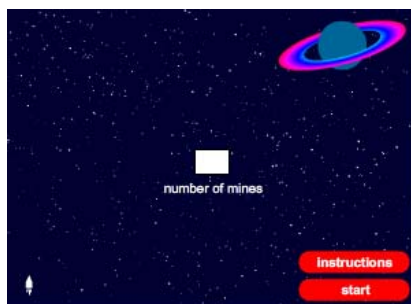
## Pas 18 de 22

Perquè el bloc de text no ens quedi massa gran, podem escriure-hi dos nombres de prova, i ajustar llavors la mida del camp.

Una vegada adaptada la mida, esborrem els nombres que havíem escrit com a guia i centrem el camp de text en l'escenari.



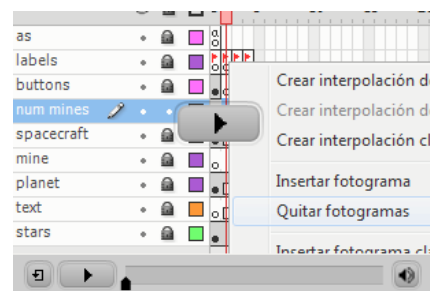
En la mateixa capa, creem un camp de **text clàssic estàtic** sota el camp d'introducció de text, i escrivim el text *number of mines* en color blanc.



En la mateixa capa també podem incloure una instància de *mine*, perquè es mostri sobre el camp de text. Per fer-ho, l'arrosseguem directament des de la biblioteca.



Perquè aquests elements només es mostrin en el primer fotograma, seleccionem la resta de fotogrames d'aquesta capa, fem clic amb el **botó dret** del ratolí, i seleccionem **Quitar fotogrames** del menú contextual.



## Paso 19 de 22

Tornant a la programació, en primer lloc podem indicar que, abans de prémer cap botó, el focus de l'escenari hauria d'estar en el camp d'introducció de text, que havíem anomenat *mines\_txt*. D'aquesta manera podrem escriure directament en el camp de text, sense necessitat de clicar-hi prèviament.

Per fer-ho, afegirem, al costat de la definició inicial de les variables del nostre joc, la instrucció següent:

```
stage.focus = mines_txt;
```

Després, quan premem el botó *start*, el focus passarà a la nau, ja que és el que havíem indicat en la funció *playGame*. És per això que no hi ha cap problema a establir el focus inicial del joc en el camp de text.

Si ara provem el joc comprovarem que, si escrivim un nombre amb el teclat, aquest apareixerà en el camp de text sense necessitat d'haver-lo seleccionat prèviament.

Canviem el valor inicial de la variable *numMines* a 0 en comptes de 10:

```
var numMines:Number = 0;
```

Una vegada que es premi el botó *start*, el primer que s'ha de fer és assignar a la variable *numMines* el valor que hàgim escrit en el camp *mines\_txt*.

Per tant, la primera instrucció dins de la funció *playGame*, serà:

```
numMines = Number(mines_txt.text);
```

Per a accedir al contingut d'un camp de text, hem d'accedir a la propietat *text* del camp. El contingut d'un camp de text sempre és del tipus *String*. Tanmateix, la variable *numMines* és de tipus numèric. Per això, és necessari convertir el contingut textual del camp de text *mines\_txt* en un nombre utilitzant *Number*.

Tornem a provar el nostre joc. El nombre de mines es correspondrà al que indiquem en el camp *mines\_txt*. Tanmateix, el joc també començarà si deixem el camp en blanc o si hi introduïm altres caràcters no numèrics. El pas següent serà controlar aquestes possibles circumstàncies.

## Pas 20 de 22

En la funció *playGame* podem indicar que, una vegada que tinguem el *numMines* a partir del text introduït en el camp de text, s'avalui el nombre de mines.

Si s'introdueix un nombre de mines entre 1 (que hi hagi almenys una mina) i 50 (nombre amb el qual és molt difícil arribar al planeta), llavors que s'executin totes les altres instruccions de la funció *playGame*.

Per tant, en la funció *playGame* primer hi haurà la línia

```
numMines = Number(mines_txt.text);
```

Després hi haurà un condicional que avaluarà si *numMines* és dins del marge que considerem vàlid, i, si és així, s'executarà la resta de la funció i el joc començarà.

```
if (numMines > 0 && numMines < 51)
{
    /*put here the rest of the statements we have in PlayGame
function and start the game*/
}
else
{
    /*what do we do if numMines is out of bounds?
will display an error message*/
}
```

Per al missatge d'error, podem dibuixar un rectangle vermell similar al dels botons, i escriure-hi un text d'avís com per exemple *enter a number of mines between 1 and 50*.

Seleccionem el rectangle i el text d'avís, i premem **F8** per convertir-lo en un **clip de pel·lícula** anomenat *alert*. Li donem el nom d'instància *alert\_mc*.



A l'inici de la pel·lícula, aquest clip ha de ser invisible, per la qual cosa, juntament amb la definició de variables a l'inici de la programació, escriurem

```
alert_mc.visible = false;
```

Si *numMines* no està dins dels límits, volem que es mostri l'avís. A més, podem afegir instruccions per a esborrar el nombre erroni que hem introduït, i també per a tornar el focus al camp de text, ja que s'haurà perdut en prémer el botó *start*.

Per a tot això escriurem entre les claus de l'*else*

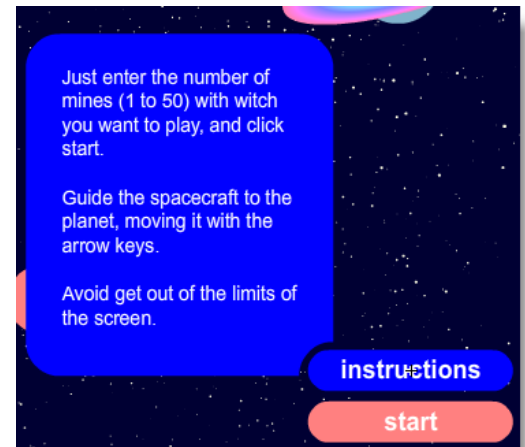
```
alert_mc.visible = true;
mines_txt.text = "";
stage.focus = mines_txt;
```

## Paso 21 de 22

En cas que `numMines` estigui dins del marge vàlid, la pel·lícula es desenvoluparà en el fotograma *game* (fotograma 2), en què ja no hi ha l'avís; per això en aquest cas no és necessari indicar que `numMines` no està visible.

Per acabar, ens queda modificar la informació que mostra el botó *instructions*, perquè també hi aparegui informació sobre les mines. Possiblement haurem d'augmentar la mida del rectangle sobre el qual apareixen les instruccions.

Amb això ja hurem completat la programació del nostre joc, amb el qual hem après a programar moltes tasques habituals en els jocs, com ara controlar objectes amb el teclat, detectar col·lisions, afegir elements des de la biblioteca, generar atzar, etc.



## Pas 22 de 22

Per a complementar els conceptes desenvolupats en aquest tutorial, es recomana fer les activitats següents:

1. Permet al jugador seleccionar la velocitat de la nau i la vibració de les mines.
2. Crea nivells per al joc, de tal manera que en arribar al planeta es comenci un joc nou amb més mines, més velocitat i més vibració.
3. Afegeix un nou control per al moviment de la nau, fent que la barra espaiadora n'aturi i repregui el seu moviment.

